Roman Parkhomenko

CST-221

File Input/Output

06/22/2019

**The Four Software Layers**

For an I/O device to communicate with the hardware in a modern computer, there are typically 4 software layers in the operating system that perform the necessary tasks to make input/output possible.

1. **Interrupt Handlers**

The lowest level of I/O software layers is the interrupt handler. The CPU needs to be able to tell when a signal has been sent from an input device. To read the signal, the CPU can use a couple of techniques called polling or interrupts. Polling requires the CPU to periodically send a signal to see if any input is waiting to be processed. This is an inefficient use of resources, so the interrupt handler is more frequently used. An interrupt is a signal sent by a device controller, which is an electrical component on the I/O device, via bus to the CPU. This interrupt signal notifies the CPU that input needs to be processed. When the CPU processes this input, it uses an interrupt handler to store it’s existing state and data structures and moves them into a specific memory address. Linux systems specifically use advanced programmable interrupt controllers (APIC’s) that allow the interrupt handler to select it’s interrupt and save routines programmatically.

1. **Device Drivers**

In the second I/O software layer are device drivers. Device drivers are small pieces of software that usually come from the manufacturer of a given I/O device. The purpose of these drivers is to encapsulate specific functionality of a device and allow it to communicate with the hardware and operating system. Device drivers create a common interface, whether it’s block or character signals, for specific device I/O signals to be processed by the CPU. A device driver will read requests from any device-independent software that needs access to the I/O device, and then send those requests to the device controller or perform any necessary error handling. In Linux specifically, many device drivers can be installed onto the system to handle a variety of different inputs. For example, if I wanted to use a Logitech keyboard, I could install the necessary device drivers from Logitech so that my Linux system could handle keyboard I/O.

1. **Device-Independent Software**

Device-Independent software is the third software layer for I/O handling. As mentioned in the previous layers, there’s a common interface that allows device drivers to communicate with system hardware and the operating system. Device-independent software is that interface. Basically, this is software that creates common functions for I/O operations such as storage allocation, error handling, device protection, and device naming. This interface lives in the kernel and makes it possible to communicate common I/O signals to the hardware. On Linux systems specifically, there exists a /dev directory in the filesystem that contains a list of all connected I/O devices and their specific information.

1. **User Level I/O Software**

The user level or user space I/O software layer is the layer that’s directly interacted with by the user of an operating system or I/O device. This layer consists of libraries and interfaces that make it possible for input signals to be meaningful when routed to the other software layers. This level of software is what interprets different program execution, word processor programs, and even terminal commands. In Linux, we can see this when executing commands in the terminal, or in the C programs we write that use commands such as printf or scanf.

**Interrupt Handler**

When an I/O device needs to send a signal to the CPU to process input, the CPU calls an interrupt handler. This handler will typically take the current state and data structures in the CPU and send them to a memory block. Once the CPU state is saved, the handler will determine the type of interrupt signal that was sent and then choose an appropriate routine to route the input signal. Once the CPU handles the input signal, the interrupt handler will return any necessary data structures and CPU state back and resume the processes that were interrupted.

**Keyboard I/O**

The keyboard is an input device that allows a user to input specific commands and actions. Each keyboard contains a series of switches and circuits that send a signal to the device controller. The device controller then uses the manufacturers device driver to communicate with the device-independent layer. This layer contains a keymap which can read exactly which signal was sent by the mechanical device and can be then interpreted on screen. I would imagine a keyboard uses a character driver instead of a block driver, since each keypress could indicate a different character or action sent from the device.

**Mouse I/O**

The mouse probably works in a similar way to the keyboard in that it is an input device that uses a device driver to communicate movement to the device-independent software layer. The biggest difference that I see between these devices, is that due to the speed that a mouse moves at, it would use a block driver to send all of the pixel locations that the mouse visited in a short period of time, and then draws those locations on screen to show the mouse moving. Each mouse click also sends a signal via the device controller the same way which is then routed to the device-independent software and the user layer as well.

